An Adaptive Large Neighborhood Search-based Three-Stage Matheuristic for the Vehicle Routing Problem with Time Windows

Christensen, Jonas Mark; Røpke, Stefan

Publication date:
2015

Document Version
Peer reviewed version

An Adaptive Large Neighborhood Search-based Three-Stage Matheuristic for the Vehicle Routing Problem with Time Windows

Jonas Christensen[1], Stefan Ropke[2]

[1] Department of Transport ; Technical University of Denmark ; Denmark
jomc@transport.dtu.dk
[2] Department of Management Engineering ; Technical University of Denmark ; Denmark
ropke@dtu.dk

Keywords: *Vehicle Routing Problem, Large Neighbourhood Search, Matheuristic, Set Cover Problem, Operations Research.*

1 Introduction

The Vehicle Routing Problem with Time Windows (VRPTW) consist of determining a set of feasible vehicle routes to deliver goods to a set of customers using a hierarchical objective; first minimising the number of vehicles used and, second, the total driving distance.

A three-stage method is proposed for the VRPTW. The first stage aims at minimising the number of vehicle used, whereas the second and third phase aims at minimising the travel distance. The first stage maintains an ejection pool with temporarily unserved customers, and tries to insert these customer into the existing solution. If a new candidate solution for the minimum number of vehicles is found, a route is removed and the customers from the route is placed in the ejection pool. If the heuristic terminates without finding a new candidate solution where all customers are served, the first stage returns the last found candidate solution that serves all the customers. The second stage uses an Adaptive Large Neighborhood Search (ALNS) algorithm to minimise the travel distance, during the second phase all of the generated routes are considered by solving a set cover problem. The ALNS algorithm uses 4 destroy operators, 2 repair operators and a local search method. The third stage is a Branch-Cut-and-Price (BCP) matheuristic which considers a reduced problem instance, with only a small subset of the total arcs. This work contributes to the existing literature by integrating a set cover model with ALNS and by using BCP as a post-optimization procedure.

2 Solution Method

Each stage of the method can be seen as separate algorithms that, when combined yields the proposed three-stage method. The first phase takes as input a feasible solution and tries to minimise the number of routes in the solution. The second phase also takes as input a feasible solution and minimises the travel distance while making sure that the number of routes used in the input solution

is not exceeded. The third stage takes a subset of the arcs as input and solves this new, smaller, VRPTW instance to optimality using Branch-cut-and-Price. The following subsections explains each of the phases in more details.

2.1 Phase I

The first phase starts by removing a route from the initial solution, and places the customers in an ejection pool, which throughout phase I will contain the customers that are unserved in the current solution. If no customers remains in the ejection pool, then a new feasible solution has been found, and the number of routes used are decreased by one, in which case the procedure restarts. Phase I is an adaption of the procedure to minimize the number of routes described in [5].

2.2 Phase II

Phase II uses Adaptive Large Neighborhood Search [5] to minimise the distance travelled. In each iteration the algorithm adaptively chooses between 4 destroy operator, and 2 repair operators, after which the solution is destroyed and repaired using the chosen operators. Hereafter a simple local search method is used if the attempted solution is promising. The 4 destroy operators are Random Removal, Worst Removal, Related Removal, and Route Removal. The first three are similar to ones described in [2], and Route Removal simply removes a random route. The repair methods are a basic greedy insertion and a 2-regret insertion heuristic. Both these methods are described in more details in [2]. The local search method uses a simple relocate neighborhood, and tries to relocate customers to a cheaper position in the solution. The stopping criteria used in the second phase is based on the number of iterations.

The second phase maintains a set R with all the routes encountered, to be considered through a set cover problem (SCP). The SCP is solved every t'th iteration with the intend to redirect the search towards new unexplored parts of the solution space. The SCP can be stated as follows

$$\min \left\{ \sum_{r \in R} \bar{c}_r x_r | Ax \geq 1, x \in \{0,1\} \right\} \tag{1}$$

Where x is a binary decision variable that describes if a route is used in the solution. Here $\bar{c}_r = M + c_r$ where the c_r is the length of route r. Adding the sufficiently large number M puts significance on minimising the number of routes. If a customer is overcovered in the solution it can simply be deleted from routes until it is no longer overcovered, thus making the solution feasible again.

2.3 Phase III

The third phase is used to possibly further improve the solution found at the end of the second phase. The third phase needs a set of arcs as input for the Branch-Cut-and-Price (BCP) solver. To find a small set of 'good' arcs a set S is maintained throughout the second phase. S contains the n best unique solution found. The arcs used in the n best solution must be relative 'good' arcs, and by giving this set of arcs to the BCP-solver it is ensured that the BCP method cannot terminated with an optimal solution that is worse than the solution already found.

The VRPTW instance defined on the sparse graph is passed to a state-of-the-art BCP solver [4] which is allowed a limited time to find an improved solution.

Preliminary results shows that the BCP method is fast for the instances with 100 customers, and also efficient for the instance with 200 customers. For the bigger instances it seldom succeeds in finding an improved solution within the timelimit (600s). A major obstacle is that the pricing problem in the BCP algorithm becomes too difficult to solve for the larger instances. To alleviate this we are planning to only solve the pricing problem using heuristics.

3 Preliminary Results

An algorithm consisting of only phase 2 and 3 of the proposed algorithm, has been tested on the 56 Solomon benchmark instances [6]. The Solomon benchmark instances are divided in 2 classes, `1` and `2`, each with three subclasses `C`, `R`, `RC`. The instances from the class 1 have tight time windows, and the instances from the class 2 have wide time windows. The customers in the `C` subclass are clustered, whereas they are placed randomly and random-clustered in the `R` and `RC` classes respectively. This algorithm only focus on minimizing the total travel distance. The results are compared with known optimal solutions, as reported in [3] and [4]. The algorithm has been executed 10 times for every instance. Table 1 shows the preliminary results

	Best found solution		Average solution					Best found solution		Average solution			
	CTD	$\bar{x}^b$	CTD	$\bar{x}$	$\bar{t}$	Opt		CTD	$\bar{x}^b$	CTD	$\bar{x}$	$\bar{t}$ (s)	Opt
`C1`	7440.30	0.00%	7440.30	0.00%	31	90/90	`C2`	4699.00	0.00%	4699.00	0.00%	52	80/80
`R1`	14085.20	0.01%	14094.02	0.09%	150	82/120	`R2`	9602.40	0.05%	9624.85	0.30%	114	35/110
`RC1`	10676.70	0.01%	10683.41	0.08%	239	46/90	`RC2`	8005.80	0.01%	8016.08	0.14%	62	54/80
`All`	54509.40	0.01%	54557.66	0.11%	110	387/560							

Table 1: *Overall solution statistics. CTD is the cumulative travel distance. $\bar{x}^b$ is the gap for the best solution, and $\bar{x}$ is the gap for the average solution. $\bar{t}$ is the average time spent in seconds, and Opt denotes out of how many optimal solutions were found for each of the instance classes. Distances are truncated to one decimal as it is common in the literature on exact methods for the VRPTW.*

The preliminary results shows that the algorithm is very good at minimizing the travel distance, and on average finds solutions that are 0.11% from the known optimal solution. Moreover the algorithm finds the optimal solution for 49 out of 56 instances. Table 2 shows how the different parts of the algorithm contributes to the overall results. Here 3 different algorithm have been tested;

1. **ALNS**: Only ALNS, the set cover problems are not solved.
2. **ALNS SCP**: ALNS with the inclusion of the set cover problems. This algorithm corresponds to phase II as described in section 2.2.
3. **ALNS SCP BCP**: This algorithm uses both the second and third phase.

Table 2 shows the benefit of consider all the routes through a set cover model, as this greatly increase the quality of the solutions while incurring a reasonable increase in the time used. The BCP method increases the quality of the best solutions, and increases the number of optimal solutions found.

Figure 1 shows how the second phase converges. For this test, the stopping criteria was 25000 iterations and the set cover model was solved every 2500'th iteration. The figure shows that solving the SCP greatly helps the convergence in the beginning of the algorithm. In the end of the algorithm the effect is not as huge, but there is still a noticeable change.

		Best Solution		Average Solution		Optimals	
	$\bar{t}$	$\bar{x}$	*worst*	$\bar{x}$	*worst*	Total	Instances
ALNS	59.01	0.13%	1.64%	0.43%	2.45%	264/560	35/56
ALNS SCP	90.55	0.03%	0.54%	0.13%	0.74%	353/560	42/56
ALNS SCP BCP	109.87	0.01%	0.54%	0.11%	0.70%	387/560	49/56

Table 2: *Performance of the algorithms for the Solomon instances.* $\bar{t}$ *is the average time used for the algorithm,* $\bar{x}$ *denotes the average gap and worst is the worst achieved solution both when considering the best solution, or the average solution. The Optimal columns denotes how many optimal solutions were found, and for how many instances the algorithm found an optimal solution. The number after the '/' is the number of possible optimal solutions. Each instance were solved* 10 *times for each of the algorithms*

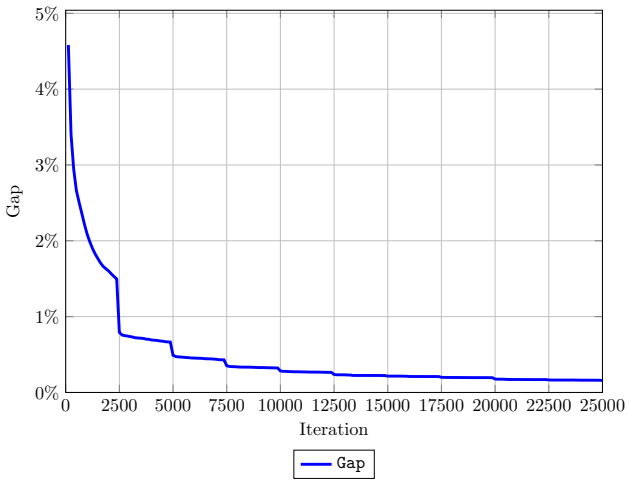

Figure 1: *Convergence of phase II*

Currently, work on finalizing and fine-tuning the vehicle minimization phase is going on. Results from this phase will be available at the the time of the presentation and the algorithm will be tested on the larger instances from Gehring and Homberger's benchmark set [1].

References

[1] Hermann Gehring and Jörg Homberger. "Parallelization of a Two-Phase Metaheuristic for Routing Problems with Time Windows". English. In: *Journal of Heuristics* 8.3 (2002), pp. 251–276. ISSN: 1381-1231.

[2] David Pisinger and Stefan Ropke. "A general heuristic for vehicle routing problems". In: *Computers and Operations Research* 34 (2007), pp. 2403–2435. ISSN: 03050548.

[3] Roberto Roberti. "Exact Algorithms for Different Classes of Vehicle Routing Problems". PhD thesis. University of Bologna, 2012.

[4] Stefan Ropke. "Branching decisions in branch- and-cut-and-price algorithms for vehicle routing problems". In: *Column Generation*. 2012.

[5] Stefan Ropke and David Pisinger. "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows". In: *Transportation Science* 40 (2006), pp. 455–472. ISSN: 0041-1655.

[6] Marius M. Solomon. "Algorithms for the vehicle routing and scheduling problems with time window constraints". In: *Operations research* 35 (1987), pp. 254–265. ISSN: 0030-364X.